Loan Application Status Prediction project

In this Dataset, we are going to solve the Loan Approval Prediction problem. This is a classification problem in which we need to classify whether the loan will be approved or not. classification refers to a predictive modeling problem where a class label is predicted for a given example of input data

Import Libraries

In [ ]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
```

Loading Dataset

In [1]:

```
import pandas as pd
df=pd.read_csv(r"https://raw.githubusercontent.com/dsrscientist/DSData/master/loan_prediction.csv")
df.head()
```

Out[1]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
0	LP001002	Male	No	0	Graduate	No	5849	0.0	NaN	360.0	1.0	Urban	Y
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	128.0	360.0	1.0	Rural	N
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	360.0	1.0	Urban	Y
3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	360.0	1.0	Urban	Y
4	LP001008	Male	No	0	Graduate	No	6000	0.0	141.0	360.0	1.0	Urban	Y

In [12]:

```
df
```

Out[12]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
0	LP001002	Male	No	0	Graduate	No	5849	0.0	NaN	360.0	1.0	Urban	Y
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	128.0	360.0	1.0	Rural	N
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	360.0	1.0	Urban	Y
3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	360.0	1.0	Urban	Y
4	LP001008	Male	No	0	Graduate	No	6000	0.0	141.0	360.0	1.0	Urban	Y
...	...	...	...	...	...	...	...	...	...	...	...	...	...
609	LP002978	Female	No	0	Graduate	No	2900	0.0	71.0	360.0	1.0	Rural	Y
610	LP002979	Male	Yes	3+	Graduate	No	4106	0.0	40.0	180.0	1.0	Rural	Y
611	LP002983	Male	Yes	1	Graduate	No	8072	240.0	253.0	360.0	1.0	Urban	Y
612	LP002984	Male	Yes	2	Graduate	No	7583	0.0	187.0	360.0	1.0	Urban	Y
613	LP002990	Female	No	0	Graduate	Yes	4583	0.0	133.0	360.0	0.0	Semiurban	N

614 rows × 13 columns

In [2]:

```
df.shape
```

Out[2]:

```
(614, 13)
```

In [3]:

```
df.columns
```

Out[3]:

```
Index(['Loan_ID', 'Gender', 'Married', 'Dependents', 'Education',
       'Self_Employed', 'ApplicantIncome', 'CoapplicantIncome', 'LoanAmount',
       'Loan_Amount_Term', 'Credit_History', 'Property_Area', 'Loan_Status'],
      dtype='object')
```

In [4]:

```
df.head()
```

Out[4]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
0	LP001002	Male	No	0	Graduate	No	5849	0.0	NaN	360.0	1.0	Urban	Y
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	128.0	360.0	1.0	Rural	N
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	360.0	1.0	Urban	Y
3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	360.0	1.0	Urban	Y
4	LP001008	Male	No	0	Graduate	No	6000	0.0	141.0	360.0	1.0	Urban	Y

In [6]:

```
df.tail()
```

Out[6]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
609	LP002978	Female	No	0	Graduate	No	2900	0.0	71.0	360.0	1.0	Rural	Y
610	LP002979	Male	Yes	3+	Graduate	No	4106	0.0	40.0	180.0	1.0	Rural	Y
611	LP002983	Male	Yes	1	Graduate	No	8072	240.0	253.0	360.0	1.0	Urban	Y
612	LP002984	Male	Yes	2	Graduate	No	7583	0.0	187.0	360.0	1.0	Urban	Y
613	LP002990	Female	No	0	Graduate	Yes	4583	0.0	133.0	360.0	0.0	Semiurban	N

In [7]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 614 entries, 0 to 613
Data columns (total 13 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Loan_ID            614 non-null    object
 1   Gender             601 non-null    object
 2   Married            611 non-null    object
 3   Dependents         599 non-null    object
 4   Education          614 non-null    object
 5   Self_Employed      582 non-null    object
 6   ApplicantIncome    614 non-null    int64
 7   CoapplicantIncome  614 non-null    float64
 8   LoanAmount         592 non-null    float64
 9   Loan_Amount_Term   600 non-null    float64
 10  Credit_History     564 non-null    float64
 11  Property_Area      614 non-null    object
 12  Loan_Status        614 non-null    object
dtypes: float64(4), int64(1), object(8)
memory usage: 62.5+ KB
```

In [8]:

```
df.describe()
```

Out[8]:

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
count	614.000000	614.000000	592.000000	600.00000	564.000000
mean	5403.459283	1621.245798	146.412162	342.00000	0.842199
std	6109.041673	2926.248369	85.587325	65.12041	0.364878
min	150.000000	0.000000	9.000000	12.00000	0.000000
25%	2877.500000	0.000000	100.000000	360.00000	1.000000
50%	3812.500000	1188.500000	128.000000	360.00000	1.000000
75%	5795.000000	2297.250000	168.000000	360.00000	1.000000
max	81000.000000	41667.000000	700.000000	480.00000	1.000000

Checking the NULL values

In [9]:

```
df.isnull()
```

Out[9]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
0	False	False	False	False	False	False	False	False	True	False	False	False	False
1	False	False	False	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...	...	...	...	...
609	False	False	False	False	False	False	False	False	False	False	False	False	False
610	False	False	False	False	False	False	False	False	False	False	False	False	False
611	False	False	False	False	False	False	False	False	False	False	False	False	False
612	False	False	False	False	False	False	False	False	False	False	False	False	False
613	False	False	False	False	False	False	False	False	False	False	False	False	False

614 rows × 13 columns

In [10]:

```
df.isnull().sum()
```

Out[10]:

```
Loan_ID               0
Gender               13
Married               3
Dependents           15
Education             0
Self_Employed        32
ApplicantIncome       0
CoapplicantIncome     0
LoanAmount           22
Loan_Amount_Term     14
Credit_History       50
Property_Area         0
Loan_Status           0
dtype: int64
```

In [11]:

```
df.isnull().sum().sum()
```

Out[11]:

```
149
```

In [20]:

```
df['Dependents'].unique()
```

Out[20]:

```
array(['0', '1', '2', '3+', nan], dtype=object)
```

In [21]:

```
df['Dependents'].value_counts()
```

Out[21]:

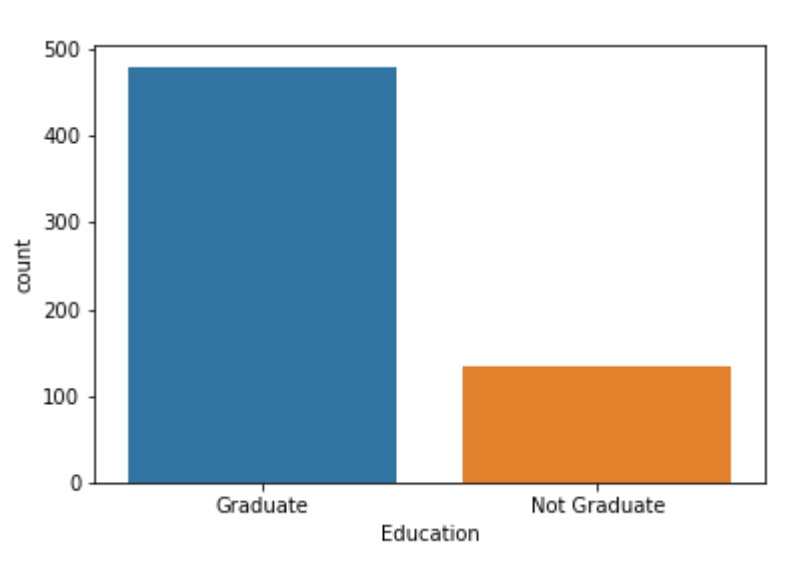
```
0     345
1     102
2     101
3+     51
Name: Dependents, dtype: int64
```

In [25]:

```
import warnings
warnings.filterwarnings('ignore')
import seaborn as sns
sns.countplot(df['Education'])
```

Out[25]:

```
<AxesSubplot:xlabel='Education', ylabel='count'>
```

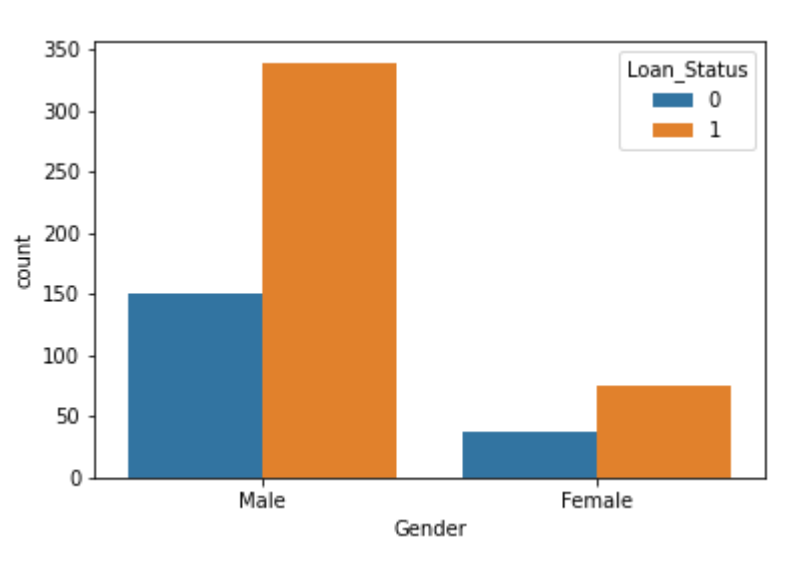


In [59]:

```
sns.countplot(x='Gender',hue='Loan_Status',data=df)
```

Out[59]:

```
<AxesSubplot:xlabel='Gender', ylabel='count'>
```



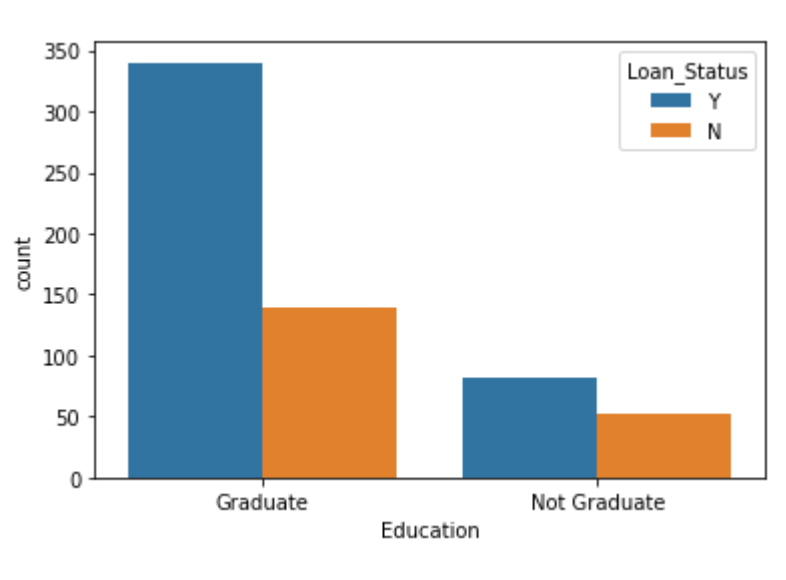
The 'Gender' column has 2 categories — Male and Female, where number of males are almost double to number of females. Missing values are not found in this column.

In [26]:

```
sns.countplot(x='Education',hue='Loan_Status',data=df)
```

Out[26]:

```
<AxesSubplot:xlabel='Education', ylabel='count'>
```



In the graph we can see that most of the people are graduate and least of the people are Non graduate.

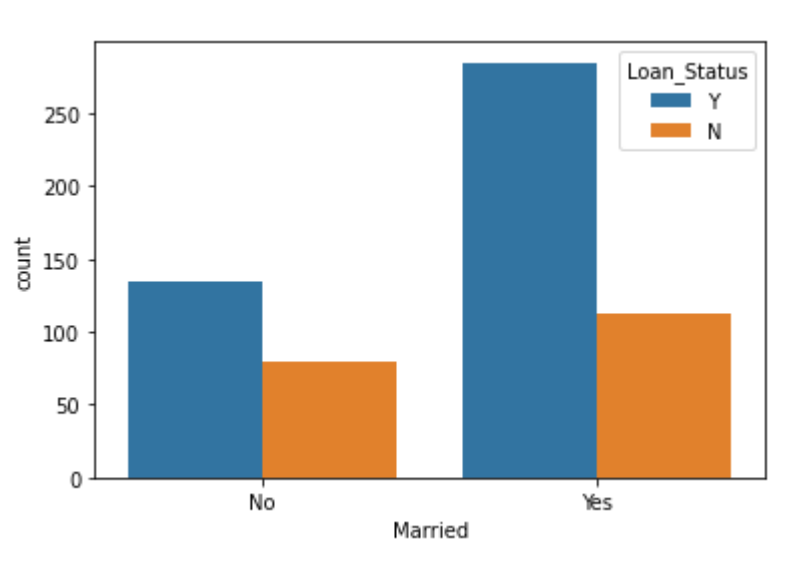
In [ ]:

In [27]:

```
sns.countplot(x='Married',hue='Loan_Status',data=df)
```

Out[27]:

```
<AxesSubplot:xlabel='Married', ylabel='count'>
```



Majority of the people have 'Marital_Status' as 'Married', and least have 'UnMarried'.

In [29]:

```
df['Married'].unique()
```

Out[29]:

```
array(['No', 'Yes', nan], dtype=object)
```

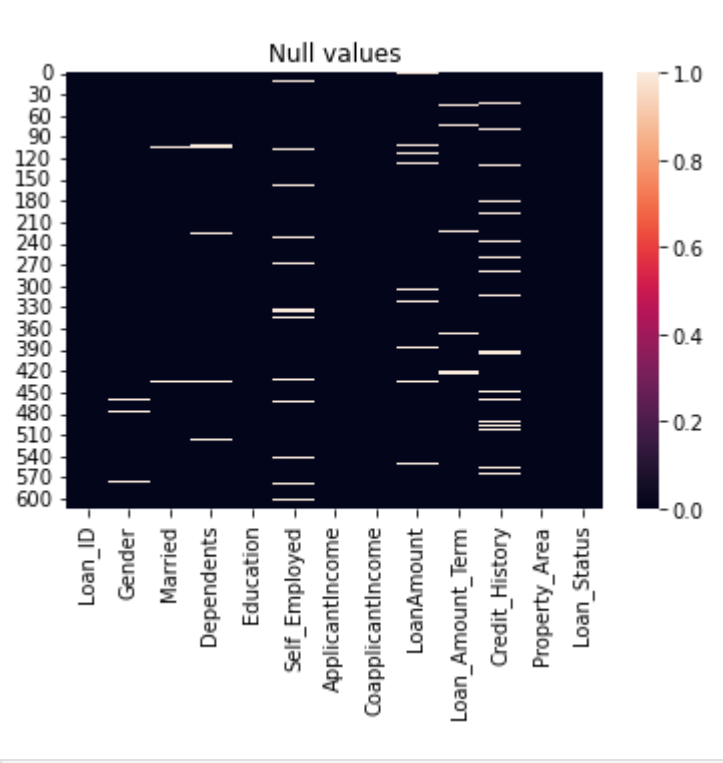
Graphical Analysis

In [54]:

```
import seaborn as sns
import matplotlib.pyplot as plt
sns.heatmap(df.isnull())
plt.title('Null values')
plt.show
```

Out[54]:

```
<function matplotlib.pyplot.show(close=None, block=None)>
```



In [30]:

```
df.head()
```

Out[30]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
0	LP001002	Male	No	0	Graduate	No	5849	0.0	NaN	360.0	1.0	Urban	Y
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	128.0	360.0	1.0	Rural	N
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	360.0	1.0	Urban	Y
3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	360.0	1.0	Urban	Y
4	LP001008	Male	No	0	Graduate	No	6000	0.0	141.0	360.0	1.0	Urban	Y

Correlation

In [55]:

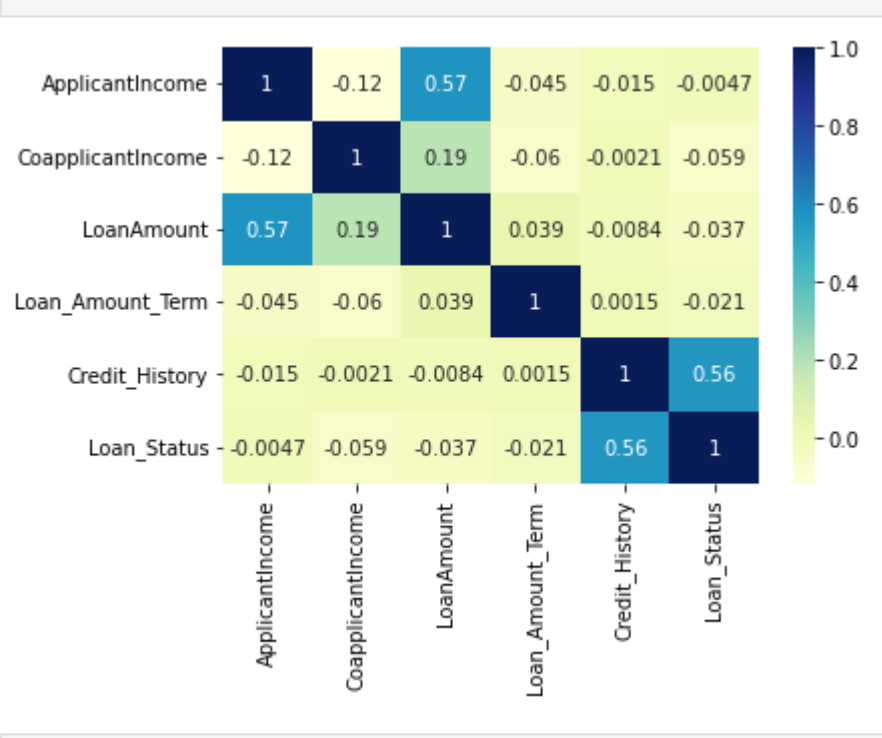
```
corr_matrix=df.corr()
corr_matrix
```

Out[55]:

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Loan_Status
ApplicantIncome	1.000000	-0.116605	0.570909	-0.045306	-0.014715	-0.004710
CoapplicantIncome	-0.116605	1.000000	0.188619	-0.059878	-0.002056	-0.059187
LoanAmount	0.570909	0.188619	1.000000	0.039447	-0.008433	-0.037318
Loan_Amount_Term	-0.045306	-0.059878	0.039447	1.000000	0.001470	-0.021268
Credit_History	-0.014715	-0.002056	-0.008433	0.001470	1.000000	0.561678
Loan_Status	-0.004710	-0.059187	-0.037318	-0.021268	0.561678	1.000000

In [51]:

```
sns.heatmap(df.corr(), cmap="YlGnBu", annot = True)
plt.show()
```



In [53]:

```
df.corr()
```

Out[53]:

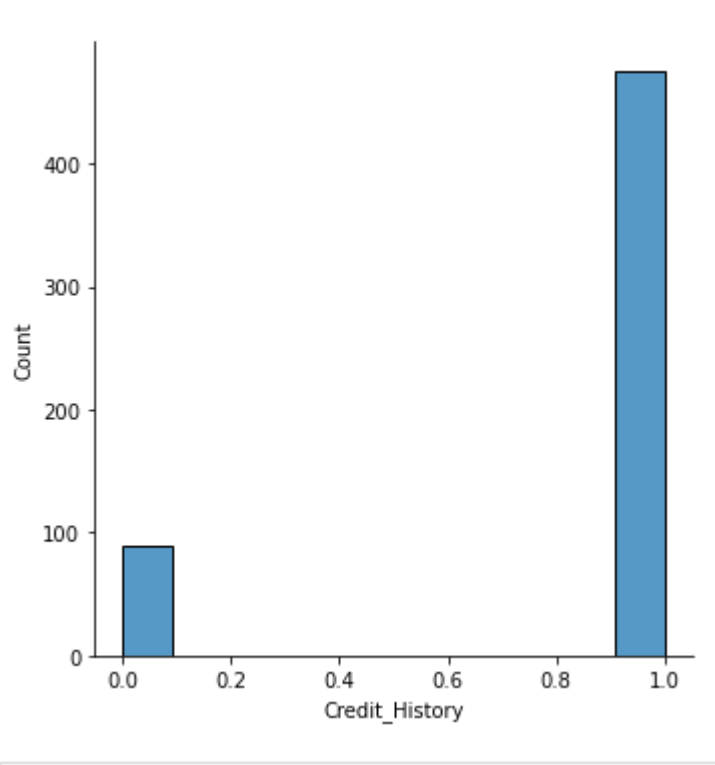
	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Loan_Status
ApplicantIncome	1.000000	-0.116605	0.570909	-0.045306	-0.014715	-0.004710
CoapplicantIncome	-0.116605	1.000000	0.188619	-0.059878	-0.002056	-0.059187
LoanAmount	0.570909	0.188619	1.000000	0.039447	-0.008433	-0.037318
Loan_Amount_Term	-0.045306	-0.059878	0.039447	1.000000	0.001470	-0.021268
Credit_History	-0.014715	-0.002056	-0.008433	0.001470	1.000000	0.561678
Loan_Status	-0.004710	-0.059187	-0.037318	-0.021268	0.561678	1.000000

In [61]:

```
sns.displot(df.Credit_History)
```

Out[61]:

```
<seaborn.axisgrid.FacetGrid at 0x1d24f675e80>
```

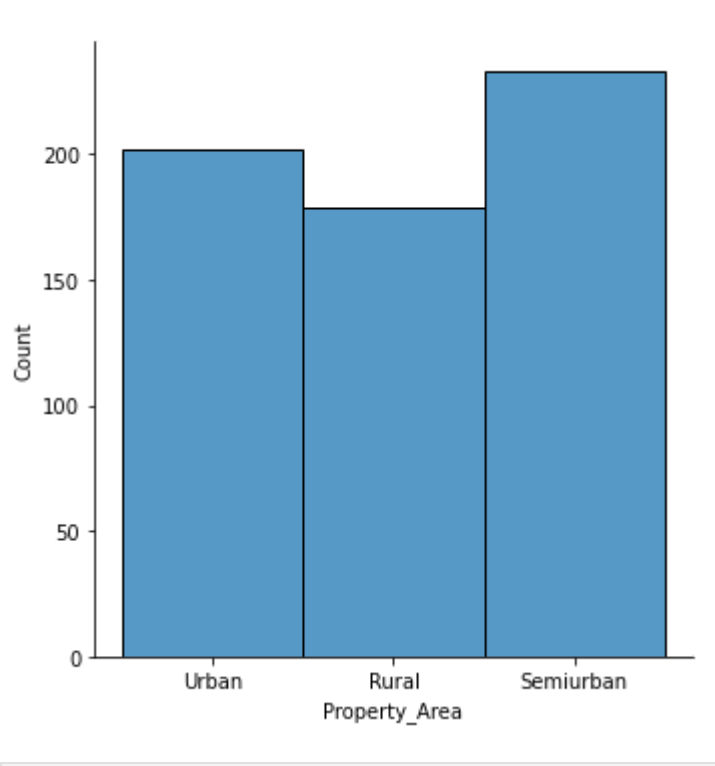


In [62]:

```
sns.displot(df.Property_Area)
```

Out[62]:

```
<seaborn.axisgrid.FacetGrid at 0x1d2503be160>
```

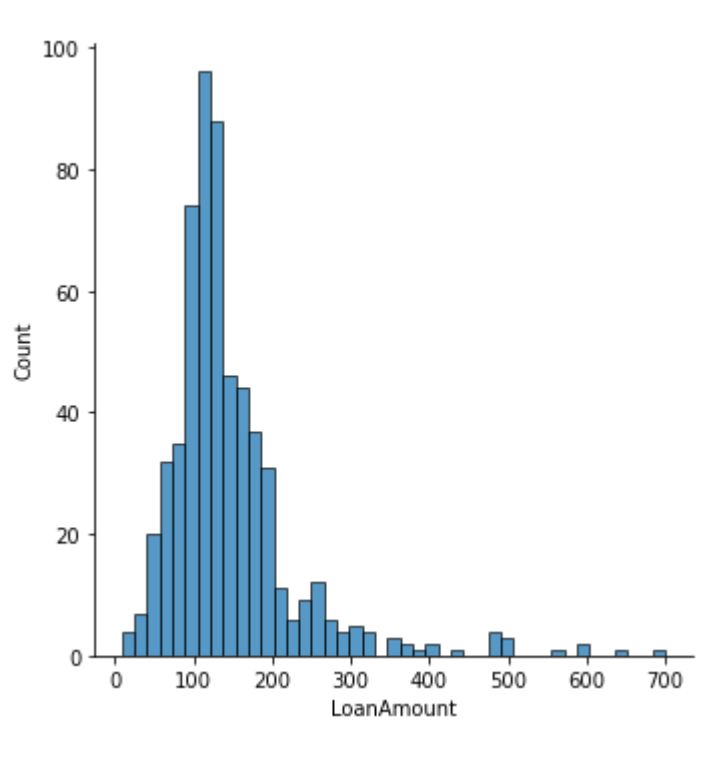


In [63]:

```
sns.displot(df.LoanAmount)
```

Out[63]:

```
<seaborn.axisgrid.FacetGrid at 0x1d2502ae700>
```

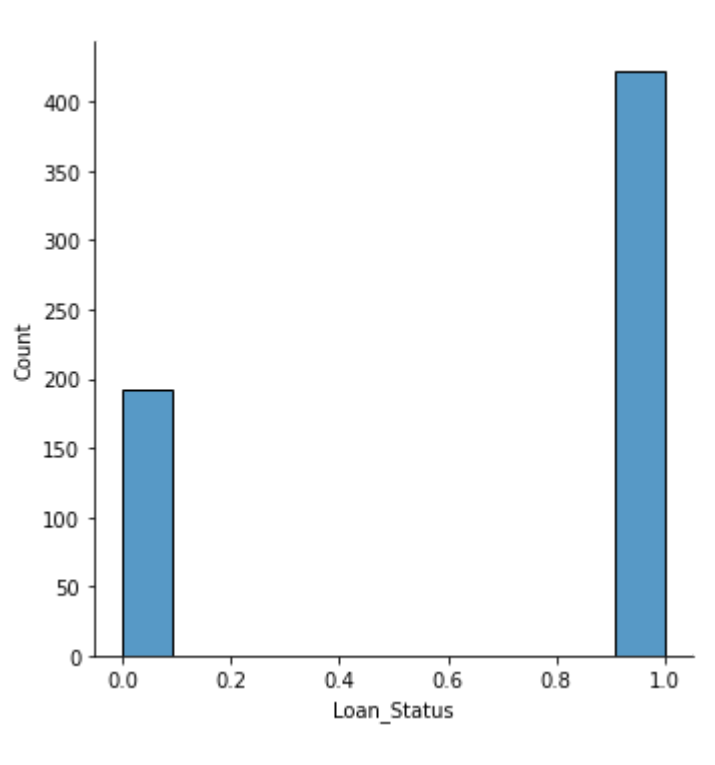


In [64]:

```
sns.displot(df.Loan_Status)
```

Out[64]:

```
<seaborn.axisgrid.FacetGrid at 0x1d2527a4880>
```



Skewness

We now proceed with treating skewness in our data, which allows us to fit our data in a symmetric distribution, which further allows our model to learn better.

The skewness of the data without any transformation is

In [65]:

```
df.skew()
```

Out[65]:

```
ApplicantIncome      6.539513
CoapplicantIncome    7.491531
LoanAmount           2.677552
Loan_Amount_Term    -2.362414
Credit_History      -1.882361
Loan_Status         -0.809998
dtype: float64
```

In [ ]:

```
from sklearn.preprocessing import Encoder
```

In [ ]:

```
le=LabelEncoder()
```

In [66]:

```
df.columns
```

Out[66]:

```
Index(['Loan_ID', 'Gender', 'Married', 'Dependents', 'Education',
       'Self_Employed', 'ApplicantIncome', 'CoapplicantIncome', 'LoanAmount',
       'Loan_Amount_Term', 'Credit_History', 'Property_Area', 'Loan_Status'],
      dtype='object')
```

Conclusion

We moved step by step, analyzing, cleaning and modeling the data, and applied various machine learning models to achieve the desired predictions. We also tuned the model to improve the accuracy, and were able to achieve a model with quite a good accuracy.

THANK YOU